GMPLS/Lightwave Agile Switching Simulator

Oliver Borchert

GLASS January Workshop

January 6th to January 10th 2003

Software – www.antd.nist.gov/glass
Questions – glass@antd.nist.gov

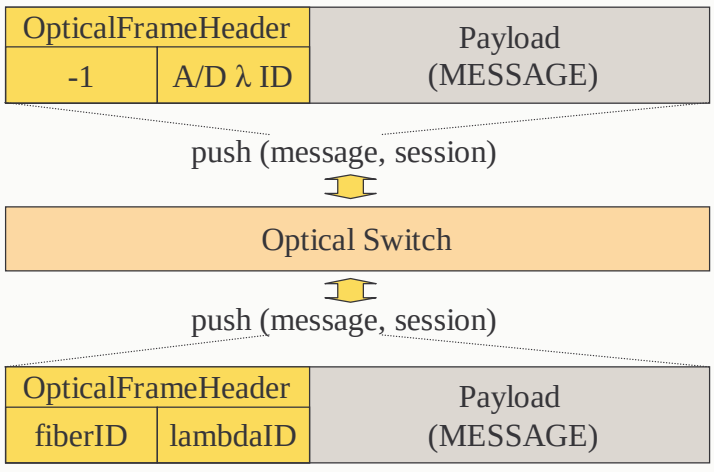
Agenda - Tuesday

- Tuesday
  - Optical Protocols
    - The OXCSwitch
    - Signaling Protocol Example
- Lunch
  - Algorithms
  - The Optical Path Structure
  - Utilities
  - The TSC Configuration

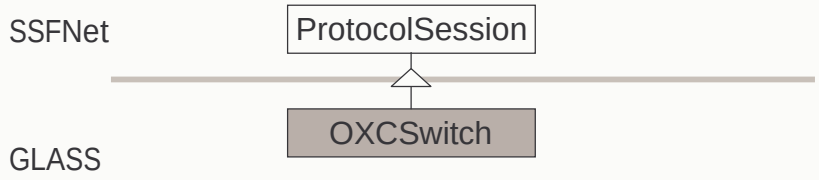
The Optical Frame Header OFH

- The OFH is a virtual header to specify the physical path the package has to use.
- The size of the header is zero bytes.
- The OFH is used by the ONIC and the Optical switch (OXCSwitch).

The OFH and the Optical Switch



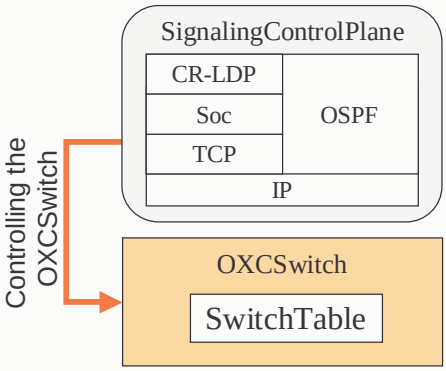
The Optical Switch class hierarchy



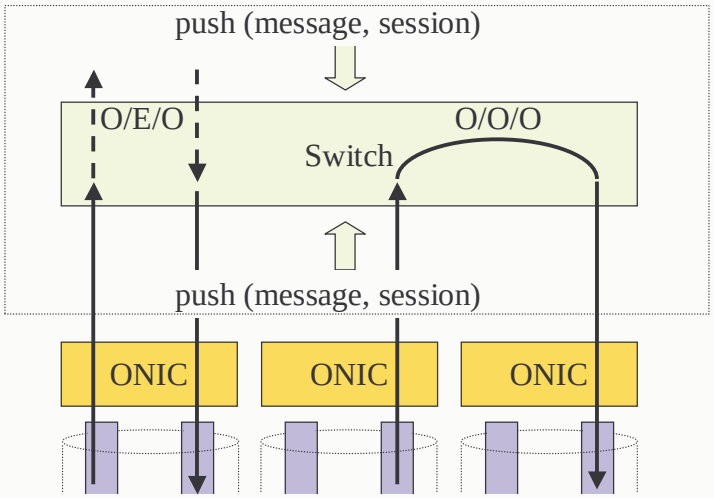
The Optical Switch

- The Optical Switch is located between the ONICs and the protocols.
- The Protocols on top need to do the Framing Adaptation.
- The Interface for sending/receiving messages is the OFH in both directions.
- The Switch provides OOO Switching as well as OE and EO conversion via Add/Drop Ports.

Controlling the Switch

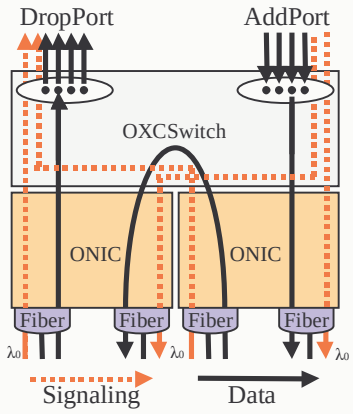


Processing of Optical Messages



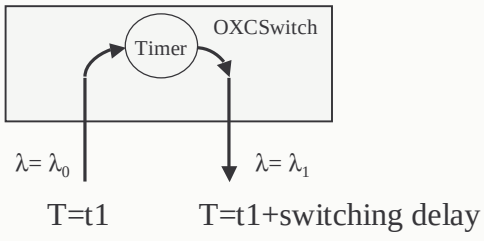
The Optical Switch Add and Drop Capability

- The Add/Drop capability allows packet switching.
- The incoming and outgoing message have the same format - The OFH.
- The Switch allows only valid connections.
- Number configurable by DML.



The Switching Delay

- Feature coming soon!
- Used to introduce delay when lambda conversion is necessary (an input lambda that has a wavelength different from the output lambda).
- Configurable by DML.



The Setup Delay

- Used to simulate the hardware response time.
- When using the method connect/disconnect.
  - 2 possible calls to these methods,
  - Public double connect (inputL, outputL)

    A protocol using this method should implement the processing delay on its own. The return value is the setup delay.
  - Public int connect (inputL, outputL, ProgramPtr)

    A protocol calling this method uses the internal processing delay offered by the switch. It must implement the interface ProgramPointer. After the setup delay, the method callback is called. The return value is a unique ID to identify the call.

Interfacing with the OXCSwitch (1)

- Control of the switch:
  - Set-up of the switching table (use of connect/disconnect).
  - Set the lambda conversion capability (also configurable by DML).
  - Add/remove ADL.
  - Retrieve status information.

Interfacing with the OXCSwitch (2)

- Message flow:
  - Before sending through the OXCSwitch, connect an ADL and a lambda.
  - The set-up of the ADL can be done by DML or dynamically by a signaling protocol.
  - For data path, the protocol must also be register to the path (control path are point-to-point).

Example

- Using the ProgramPointer interface

Example

- Using the delay-return-value by implementing the processing delay in the affected protocol.

Utilities for ADL (1)

- Abstract Class AbstractAddDropConfigurator
  - For protocol that interface with OXCSwitch
  - Provide configuration of ADL via DML file

```
addLambda [
  id       <integer> # The id of the Add/Drop Lambda (ADD Port).
  onicId   <integer> # The id of the onic.
  fiberId  <integer> # The id of the fiber the lambda is configured for.
  lambdaId <integer> # Must be a receiver lambda in this node
]
dropLambda [
  id       <integer> # The id of the Add/Drop Lambda (DROP Port).
  onicId   <integer> # The id of the onic.
  fiberId  <integer> # The id of the fiber the lambda is configured for.
  lambdaId <integer> # Must be an emitter lambda in this node
]
```

Utilities for ADL (2)

- Abstract Class AutoConfigCtrl
  - Extends AbstractAddDropConfigurator
  - In addition to DML configuration of ADL, provides autoconnection of control lambdas to and from this protocol.
  - Usefull for signaling protocol that needs to have a control lambda.
  - The algorithm connects the input lambdas to the drop ports and the output lambdas to the add ports.
  - The information (ADL Ids) is stored and accessible by methods.

ADL configuration (Example)

- 2 protocols are using these utilities:
  - DynRecovery: signaling protocol for path recovery. Package gov.nist.antd.merlin.protocol.signaling
  - Neighbor discovery: simple implementation of a neighbor discovery for optical network. Package gov.nist.antd.merlin.protocol.discovery
- Annex X shows manual configuration of ADL for the discovery protocol (also available in directory GLASS_HOME\examples\optical).

Example

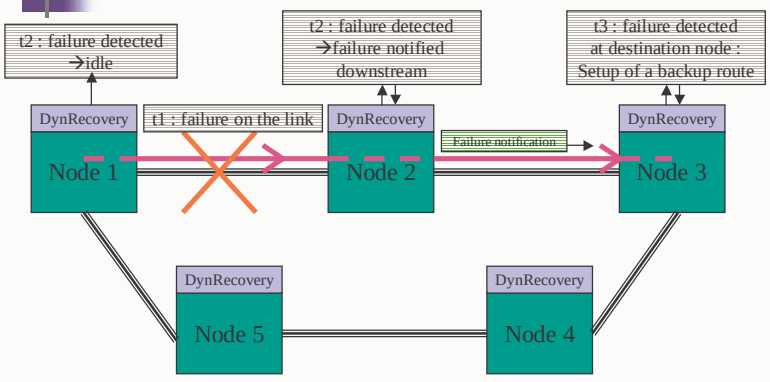
- Connecting Add Drop Lambdas and sending a message.

Protocols

Signaling Protocol

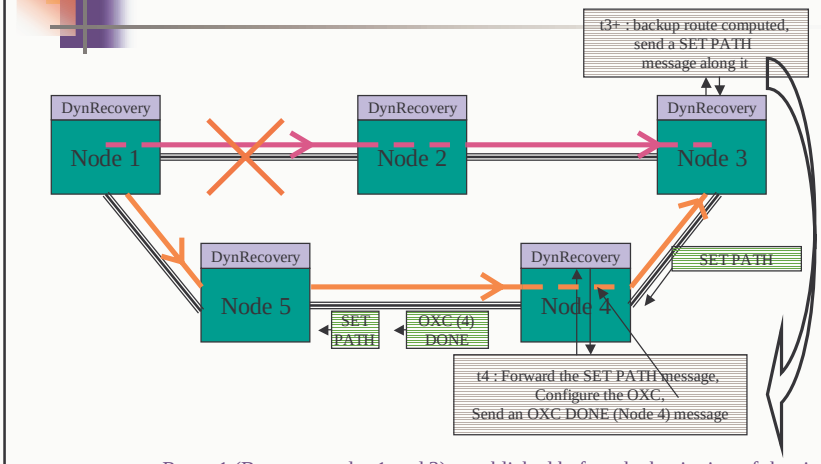
- The DynRecovery is a signaling protocol for backup route.
- Package gov.nist.antd.merlin.protocol.signaling.
- Implemented as an example (does not support all network configurations).
- The backup is computed dynamically when a failure is detected.

Signaling Protocol (2)



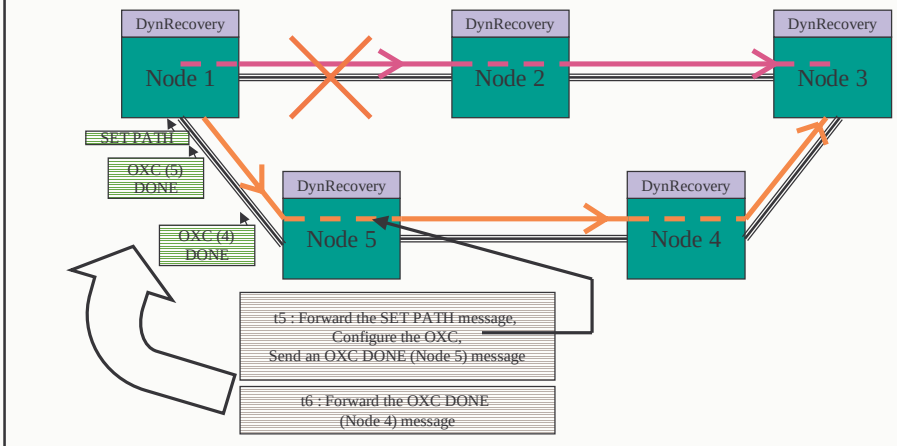
Route 1 (Between nodes1 and 3), established before the beginning of the simulation

Signaling Protocol (3)

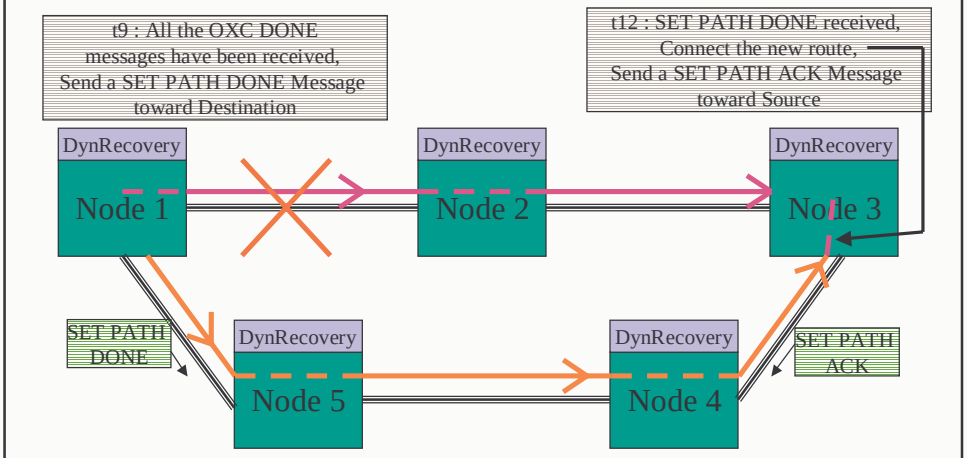


Route 1 (Between nodes 1 and 3), established before the beginning of the simulation.
Route 2 (Between nodes 1 and 3), computed during the simulation at destination node.

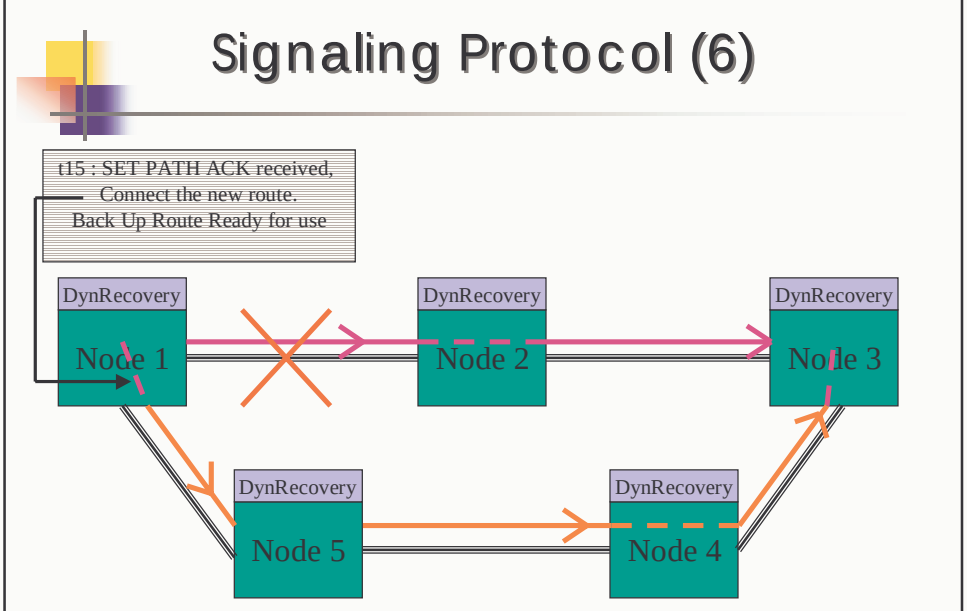
Signaling Protocol (4)



Signaling Protocol (5)

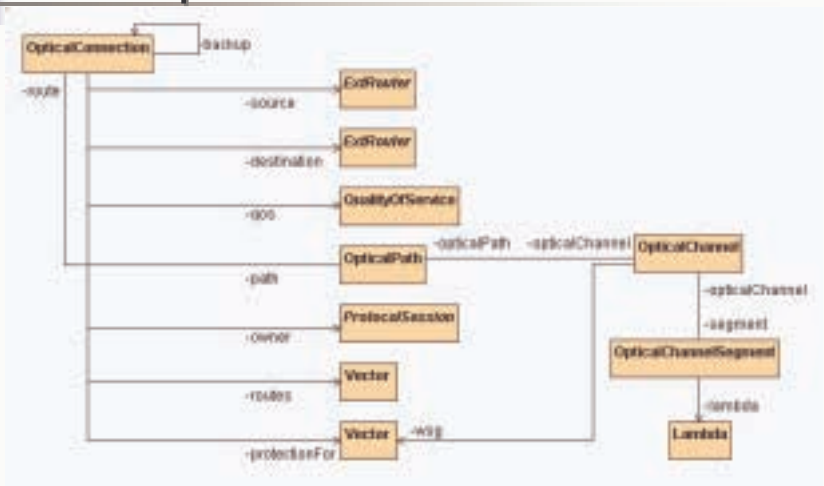


Signaling Protocol (6)



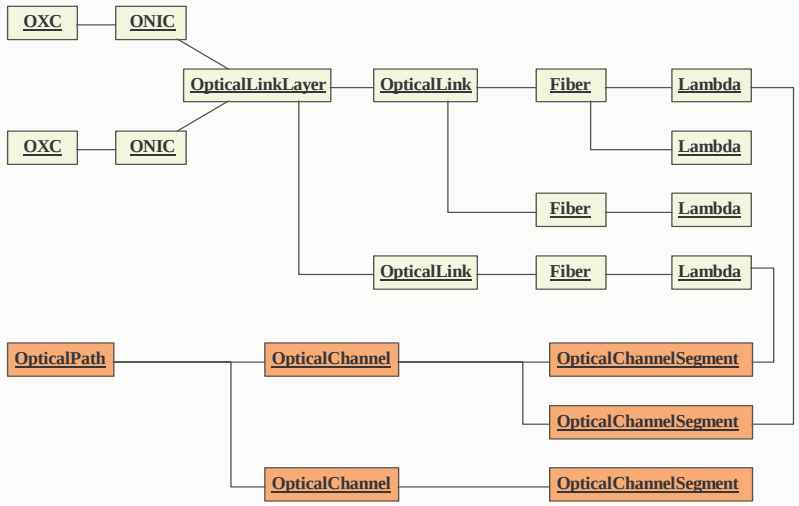
Algorithms

Internal Data Structure for optical connections



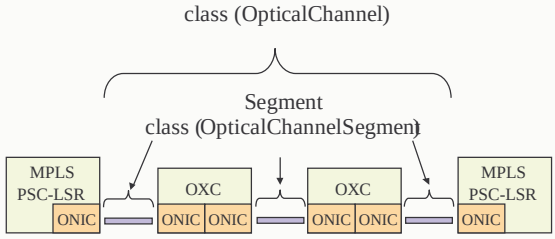
NIST National Institute of Standards and Technology

The Optical Path Mapping



NIST National Institute of Standards and Technology

The Optical Channel



Using Omniscient algorithms in the simulation

- GLASS provides mechanisms to establish a light path without using a signaling protocol.
- This allows the writer of a data protocol to focus on his problem without studying the lower layer protocols.
- Usable only with one timeline configuration.

Generic Algorithm Interface

- A writer of an algorithm does not need to know the details of the simulator.
- Use the simulation framework as topology database.
- Allow to use static algorithms in Protocols by providing a standardized interface.
- 3 Kinds of algorithms:
  - Routing only.
  - Wavelength Assignment (normally as signaling protocol).
  - RWA algorithms (combination of the previous two).

The Routing Algorithm

- Precondition:
  An existing instance of OpticalConnection is needed, where the algorithm adds the route(s).
- Postcondition
  An existing instance of OpticalConnection with a list of possible routes or an empty route list.

The Wavelength Assignment

- Precondition:
  An existing instance of OpticalConnection with a list of possible routes.
- Postcondition
  An attached instance of OpticalPath with the calculated channels. If no channels are found, the algorithm must not add a path instance.
  For each generated channel the switches in the OXC may be configured.

Algorithm interface

- The method *config(Configuration cfg, Glass net)* allows a self configuration of the algorithm via DML file.
- 2 methods can be used for the execution of an algorithm:
  - *Vector execute(Glass net, Vector routes, Vector parameter)*
  - *Object[] execute(Glass net, OpticalConnection[] routes, Object[] parameter)*

Storage

- Each algorithm is stored in a container (Class AlgorithmContainer). This allows to find algorithms dynamically.
- Each connection should be stored in a container (Class AlgorithmContainer).
- By default, the name of the Routing algorithm is used to classify the connections.
- A container “static” is created to store the static connections created via DML file.
- Utilities exist to find connections.

Why using this structure?

- One could use another structure to store the path but the TSC would not be able to find them.
- The OpticalPath object is sending events to the TSC.
- The TSC allows one to add, remove, and configure algorithms.
- Allows also the creation of connections before the simulation starts.
- Displays all the connections selected by the user.

How To Do Creation of the light path

- Define the quality of service requirement
  - Bandwidth
  - name of routing algorithm
  - name of wavelength assignment algorithm
- Request a connection by using the utilities provided in the class ConnectionUtil or by calling directly the algorithms.

One way how To Do Send a message

- The sending protocol has to register to the route it is using.
- Create an optical frame (OFH).
  - fiber ID (-1 is a valid id for the switch only)
  - lambda ID (-1 is an invalid id)
- The message is the payload of the OFH.
- Push the message into the optical layer.
  - Current receiver are the ONIC and OXCSwitch

How To Do Delete a light path

- The protocol that owns the light path can delete it.
- Use the path utilities in the class PathUtil.deleteConnection (OpticalConnection)
- Use a signaling protocol to unconnect the OXC switches along the path.

TSC configuration

- Algorithm Configuration
  - Stored in cfg/algoProfile.dml
- Connection set-up

Scripting Connections

- Example Scripting of...
  - static connection:
    - Routing and wavelength are given.
  - Semi static connection:
    - Routing is given, wavelength computed.
  - dynamic connection:
    - Routing and wavelength computed.